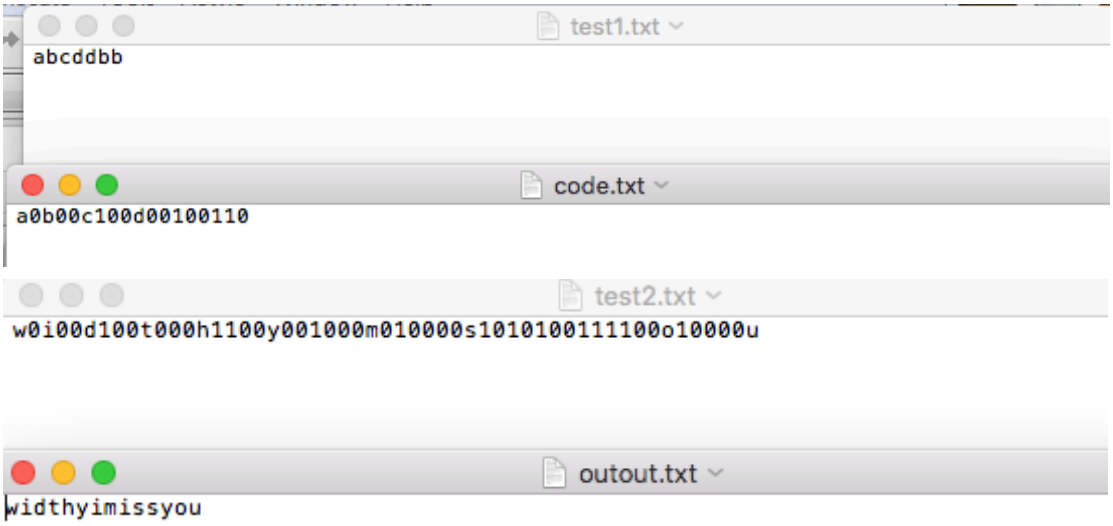

2. Describe your implementation

這次作業主要就是動態的 huffman code，在 main function 裡，先讀檔案，再來 encoding 部份就是建立 dynamic Huffman tree，先一個字元一個字元的讀，如果是樹裡面沒有的字元就讓他進入樹裡面的 NYT 裡面，再來調整樹，這個新的 node 每一次都會與和他 weight 一樣且 order 最大的做交換，調整完樹之後再來把 NYT 像左下方移，並從 root 開始如過向右輸出 1，向左輸出 0，到 NYT 的位置，如果是數裡面已經存在的字元，就把那個已經存自的字元與和他 weight 一樣且 order 最大的做交換，最後 weight 再加一，再來調整 order，並從 root 開始如過向右輸出 1
，向左輸出 0。
再來是decoding的部分，基本上幾乎跟encoding差不多，一開始先判斷獨進來的是不是數字，如果不是數字就讓他進來dynamic Huffman tree，如果是數字，從root開始如過遇到1就是向右，0就是向左，當走到當前的node 不是NYT的話就印出來再來就和encoding一樣調整整棵樹的weight和order。

3. What is the time complexity/ space complexity of the decoding/ encoding method?

time complexity :$O(n * \log|\Sigma|)$

space complexity : $O(|\Sigma|)$

4. In what case can the dynamic Huffman coding has a better compression ratio? That is, in what situation, this method can make the input file much smaller?

只要出現的頻率會可以放在越靠近 root 的地方，也就是可以把存的容量變小，所以只要越多的出現頻率的 node 出現，input file 就可以越小，compression ratio 可以變更好。